

Mapping LWM2M model to CoMI YANG

draft-vanderstok-core-yang-LWM2M-00

Peter van der Stok
Jaime Jiménez (presenting)

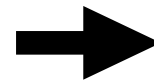
(Work in Progress)

Purpose

- Standard organisations (BACnet, KNX, ZigBee, oBIX, OMA/IPSO) define hierarchical models that can be specified in XML and describe classes with attributes and operations that can be instantiated to objects.
 - OMA LWM2M and IPSO standardise numbered object types.
- CoMI at IETF (draft-vanderstok-core-comi-09) describes a network management interface based on CoAP and YANG.
- Goal: convert a LWM2M xml-based device specification to a YANG MODULE for CoMI consumption.

Conversion Rules

LWM2M



YANG

optional /mandatory attribute	false / true statement
R, W attributes	Config parameter (False=R, True=W)
E attribute	YANG rpc
range attribute	range statement
units	units statement
Resources	Leafs on a YANG list +--ro ID3301* [instance_number] +--ro 5700 uint16
Object Instance	"instance" key attribute

IPSO Humidity Object

Object definition

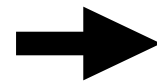
Name	Object ID	Instances	Mandatory	Object URN
Humidity	0	Multiple	Mandatory	urn:oma:lwm2m:ipso:3304

Resource definitions

ID	Name	Operations	Instances	Mandatory	Type
5700	Sensor Value	R	Single	Mandatory	Float
5601	Min Measured Value	R	Single	Optional	Float
5602	Max Measured Value	R	Single	Optional	Float
5603	Min Range Value	R	Single	Optional	Float
5604	Max Range Value	R	Single	Optional	Float
5701	Sensor Units	R	Single	Optional	String
5605	Reset Min and Max	E	Single	Optional	Opaque

URI Conversion

LWM2M



YANG

URI:

`http://example.com/type/instance/resource`

`coap+lwm2m://example.com/type/instance/resource`

RESTCONF URI (example 3):

`http://example.com/type/instance=0/resource`

CoMI URI (example 3):

`coap://example.com/type/resource?keys=0`

if only one instance then

`coap://example.com/type/resource`

- Keys as query parameter for instance number.

Generated YANG modules

		1. module: ietf-yang-humidityID
[]	list keys	+--ro ID3301* [instance_number]
		+--ro instance_number uint16
		+--ro ID5700 decimal64
rw	configuration data (read and write)	+--ro ID5701? string
		+--ro ID5601? decimal64
ro	state data (read only)	+--ro ID5602? decimal64
		+--ro ID5603? decimal64
		+--ro ID5604? decimal64
?	optional node	+---x ID5605
*	list and leaf list	
()	choice	2. module: ietf-yang-humidityNM
:	case nodes	+--ro IPSO-humidity* [instance_number]
		+--ro instance_number uint16
		+--ro Sensor_Value decimal64
		+--ro Units? string
		+--ro Min_Measured_Value? decimal64
		+--ro Max_Measured_Value? decimal64
		+--ro Min_Range_Value? decimal64
		+--ro Max_Range_Value? decimal64
...	subtrees not shown	+---x Reset_Min_and_Max_measured_values

Generated YANG modules

[] list keys

rw configuration data (read and write)

ro state data (read only)

? optional node

***** list and leaf list

() choice

: case nodes

... subtrees not shown

3. module: **ietf-yang-humidityLF**

```
+--rw IPSO-humidity
```

```
+--ro identifier      uint16
```

```
+--ro resources* [instance_number]
```

```
+--ro instance_number uint16
```

```
+--ro Sensor_Value
```

```
| +--ro identifier?   uint16
```

```
| +--ro content       decimal64
```

```
+--ro Units
```

```
| +--ro identifier?   uint16
```

```
| +--ro content?      string
```

```
+--ro Min_Measured_Value
```

```
| +--ro identifier?   uint16
```

```
| +--ro content?      decimal64
```

```
+--ro Max_Measured_Value
```

```
| +--ro identifier?   uint16
```

```
| +--ro content?      decimal64
```

```
+--ro Min_Range_Value
```

```
| +--ro identifier?   uint16
```

```
| +--ro content?      decimal64
```

```
+--ro Max_Range_Value
```

```
| +--ro identifier?   uint16
```

```
| +--ro content?      decimal64
```

```
+--ro Reset_Min_and_Max_measured_values
```

```
+--ro identifier?     uint16
```

```
+---x reset
```

Takeaways

- YANG is richer and more verbose than LWM2M.
- Multiple ways to express the same thing.
- IMO YANG still seems a bit overkill, CoMI might be able to use more purposely together with legacy devices.
- Both .XML and .YANG have a lot of “noise” in them.
- Key leafs are just one possible way to represent instances.
- Access Control mapping might not be ideal.
- Need to script automatic conversion.
- Where would a converter run? GWs, devices, server?

Links

- <http://ipso-alliance.github.io/pub/>
- <http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0>
- <http://jaimejim.github.io/drafts/draft-vanderstok-core-yang-lwm2m-00.txt>
- jaimejim.github.io/drafts/3304.xml
- jaimejim.github.io/drafts/3304.yang